Revista Digital

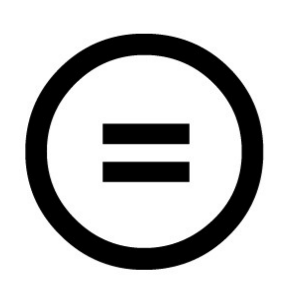
ATIX

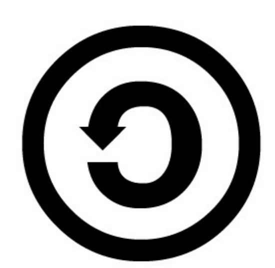
Continuamos Trabajando













2013 - Bolvia

http://revista.atixlibre.org
Twitter: @atixlibre
Facebook: facebook.com/Atix.Libre





Equipo Editorial

Dirección y Coordinación General

Esteban Saavedra López (esteban.saavedra@atixlibre.org)

Diseño y Maquetación

Jenny Saavedra López (jenny.saavedra@atixlibre.org)

Esteban Saavedra López (esteban.saavedra@atixlibre.org)

Revisiones

Esteban Saavedra López

Jenny Saavedra López

Noticias

Jenny Saavedra López

Autores Frecuentes

Esteban Saavedra López

Martín Márquez

Rafael Rendón

Ernesto Rico Smith

Francisco Ferri

Herramientas

La edición de esta revista fue realizada de forma integra haciendo uso de Software Libre







atix

El que lo Intenta

El que lo Sabe

El que lo Puede

El que lo Logra

Comunidades Trabajando

Durante los últimos meses hemos evidenciado que son varias las comunidades que han continuado fuertemente trabajando tanto a nivel nacional como internacional, una claro reflejo ha sido el último FLISOL, donde hemos observado mucha gente nueva con ganas de aprender, guiados por los miembros más antiguos con el mismo entusiasmo de siempre.

Con el título de este número deseamos expresar dos palabras de gran significado, COMUNIDADES que representa el conglomerado de personas con un interes afin, en este caso el Software, la cultura y las tecnologías libres y TRABAJO que muestra la dedicación de las personas por tan noble actividad como es la PROMOCIÓN Y DIFUSIÓN de las tecnologías, cultura y software libre en general; enmarcados en los aspectos técnicos y éticos de lo que representa en la vida y actividad de todos nosotros.

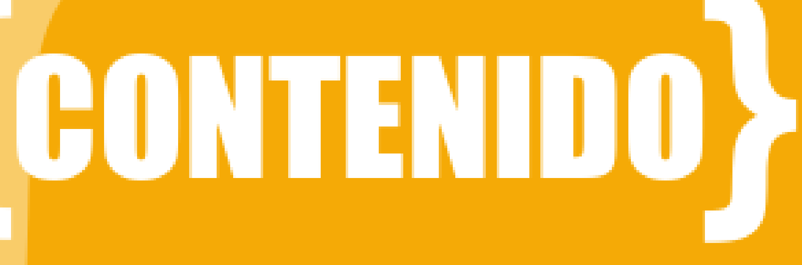
En este tercer número de este año, continuamos mostrando las diferentes alternativas de desarrollo de aplicaciones orientadas a la web, el trabajo con datos especializados en diversos ambientes y plataformas especialmente las orientadas a diversos dispositivos.

Bienvenidos a nuestro vigésimo segundo número



Esteban Saavedra López
Presidente Fundación AtixLibre

Liberado el 1 de Julio del 2013



{CONTENIDO}

Introducción a ZK y su relacion con RIA (Parte 3)

ZK es un framework de aplicaciones web en AJAX, software de código abierto que permite una completa interfaz de usuario para aplicaciones web



Declarando una Clase de Dominio

Lo siguiente es la clase de dominio que representa a un coche.

Extraído de **Car.java**

```
public class Car {
    private Integer id;
    private String name;
    private String company;
    private String preview;
    private String description;
    private Integer price;
    //omit getter and setter for brevity
}
```

Para ver el código completo de la clase puedes ir a continuación, al final del artículo, en la sección de referencias.

Ahora definimos una interfaz para implementarla en una clase de servicio que contendrá la lógica de negocio necesaria en el ejemplo, como buscar los coches.

Extraído de **CarService.java**

```
public interface CarService {

    /**
     * Retrieve all cars in the catalog.
     * @return all cars
     */
    public List<Car> findAll();

    /**
     * search cars according to keyword in name and company.
     * @param keyword for search
     * @return list of car that match the keyword
     */
    public List<Car> search(String keyword);
}
```

En este ejemplo, hemos definido la clase CarServeImpl que implementa la interfaz anterior. Para simplificar, usamos una lista estática de objetos como modelo de datos. Puedes reescribirla para que se conecte a una base de datos en una aplicación real. Los detalles de la implementación no están cubiertos en el ámbito de este artículo, puedes echar una ojeada a continuación al final del artículo, en la sección de referencias.

Creando el Interfaz de Usuario

Diseñar la interfaz de usuario es un buen comienzo para crear una aplicación, además de que te ayuda a definir el alcance de tu aplicación.

ZK nos provee de cientos de componentes, listos para ser usados en el interfaz del usuario, por lo tanto un desarrollador puede crear la interfaz de usuario que desee combinando y mezclando esos componentes sin tener que crearlos desde 0.

En ZK puedes usar el ZK Markup Language para la creación del Interfaz del usuario (ZUML) (más info al final del artículo, en la sección de referencias), que es un lenguaje estructurado como un XML, que te permite definir la Interfaz del Usuario.

La convención que se utiliza en ZK es que para los ficheros en formato ZUML utilicemos la extensión .zul como sufijo. En los ficheros zul, podemos representar un componente como un elemento del XML (tag) y configurarlo (estilo, comportamiento, funcionalidad) mediante los atributos del elemento XML. Más información al final del artículo, en la sección de referencias.

En el case de esta aplicación de ejemplo, primero, queremos diseñar una ventana con un título específico y bordes normales, como si fuera el borde exterior (marco) de la aplicación.

Extraído de **search.zul**

```
<window title="Search" width="600px" border="normal">
    <!-- put child components inside a tag's body -->
</window>
```

Como "window" es el componente que contiene al resto, lo llamamos componente raíz o “root”. El componente “window” (ventana) normalmente se usa como contenedor de otros componentes, porque en esencia, simplemente muestra una ventana vacía como en una aplicación de escritorio tradicional, y en esta podemos añadir los componentes que queramos.

Los componentes dentro de la ventana los llamaremos hijos o “child”, y deben estar dentro del cuerpo del elemento “window”.

Por ejemplo, hacemos que aparezca el título de la ventana estableciendo texto en el atributo “title” del elemento “window”, y hacemos visible el borde de la ventana estableciendo el atributo “border”. En el caso del ancho, usaremos el atributo “width”, pero en este caso estableceremos el valor de la propiedad CSS, es decir “800px” o “60%”.

Básicamente, el interfaz de usuario de nuestra aplicación de ejemplo se divide en 3 áreas dentro de la ventana (“window”), que son (de arriba a abajo): Área del buscador, Área de listado de los coches yÁrea de detalles del coche.



Figura 1.

Área del buscador

Trabajar con los componentes de ZK es como trabajar construir bloques de código, puedes combinar y mezclar componentes que existan (incluso crear los tuyos propios) para crear la interfaz de usuario que desees.

Para permitir a los usuarios buscar, necesitamos un componente que les permita escribir el texto, es decir un “input”, y un botón para lanzar la búsqueda.

A continuación vemos un ejemplo de cómo podemos usar algunos componentes simples para cubrir estos requisitos.

Extraído de **`search.zul`**

```
<hbox align="center">
    Keyword:
    <textbox id="keywordBox" />
    <button id="searchButton" label="Search" image="/img/search.png" />
</hbox>
```

hbox es un componente contenedor, que ordena horizontalmente los componentes que contenga. Seguro que has adivinado, la “h” de “hbox” significa horizontal. Como los componentes hijos o “child”, tienen diferentes tamaños, establecemos el atrubuto “align” con valor “center” para que se alineen, entre ellos sobre su línea central.

En algunos componentes del ejemplo también especificamos un atributo “id”, esto nos permitirá

referirnos a ellos y por lo tanto poder controlarlos usando el “id”. Si quieres convertir el botón en un botón con imagen solo tienes que especificar el path de la imagen en el atributo “image” del mismo.

Área de listado de los coches

ZK dispone de muchos componentes para mostrar un conjunto de datos, como por ejemplo los componentes de lista “listbox”, malla “grid” y árbol “tree”. En este ejemplo, hemos elegido usar una lista “**listbox**” para mostrar un listado de coches, con tres columnas: Nombre, Compañía y Precio.

Establecemos el atributo “height”, de forma que se mostrarán tantas filas como quepan; puedes navegar con la barra de scroll para ver el resto de filas.

El atributo “**emptyMessage**” se usa para mostrar un mensaje cuando la lista no contiene elementos.

Puesto que el componente de lista “**listbox**” también es un componente contenedor puedes añadirle un componente “listhead” para mostrar y definir los nombres de las columnas. También puedes añadirle un componente “listitem” para mostrar los datos, y dentro de este componentes “**listcell**”, tantos como columnas hayas definido (en el “**listhead**” por ejemplo).

En el siguiente código de ejemplo usamos “listcell” con contenido estático (3 listcells) para mostrar la estructura de un componente “listitem”, pero a continuación, te mostramos cómo crear un listitem dinámicamente, de acuerdo a los datos que recibe.

Extraído de **search.zul**

```
<listbox id="carListbox" height="160px" emptyMessage="No car found in the result">
    <listhead>
        <listheader label="Name" />
        <listheader label="Company" />
        <listheader label="Price" width="20%"/>
    </listhead>
    <listitem>
        <listcell label="product name"></listcell>
        <listcell label="company"></listcell>
        <listcell>$<label value="price" /></listcell>
    </listitem>
</listbox>
```

Área de detalles del coche

Al igual que el “**hbox**”, “vbox” es un componente que distribuye los componentes hijos “**child**”, pero en este caso en vertical. Combinando estos 2 componentes contenedores, podemos mostrar más información en la pantalla. El atributo “**style**” permite personalizar el estilo del componente escribiendo en él CSS directamente.

Extraído de **search.zul**

```
<hbox style="margin-top:20px">
    <image id="previewImage" width="250px" />
    <vbox>
        <label id="nameLabel" />
        <label id="companyLabel" />
        <label id="priceLabel" />
        <label id="descriptionLabel"/>
    </vbox>
</hbox>
```

Puedes ver el fichero .zul completo a continuación en la sección de referencias en al final del artículo, en la sección de referencias.

Referencias

[1] Página web oficial de ZK

http://www.zkoss.org/

[2] Guía de instalación manual de ZK y ejemplos de web.xml (Servlet 3.0, 2.4 y 2.3):

http://books.zkoss.org/wiki/ZK_Installation_Guide/Quick_Start/Create_and_Run_Your_First_ZK_Application_Manually

[3] Car.java

http://code.google.com/p/zkbooks/source/browse/trunk/tutorial/src/tutorial/Car.java

[4] Ficheros .java

CarService.java:

http://code.google.com/p/zkbooks/source/browse/trunk/tutorial/src/tutorial/CarService.java

CarServiceImpl.java

http://code.google.com/p/zkbooks/source/browse/trunk/tutorial/src/tutorial/CarServiceImpl.java

[5] ZUML:

http://books.zkoss.org/wiki/ZUML_Reference

[6] ZK Component Reference:

http://books.zkoss.org/wiki/ZK_Component_Reference

[7] Fichero search.zul completo:

http://code.google.com/p/zkbooks/source/browse/trunk/tutorial/WebContent/search.zul

Acerca de este documento

Este documento es un extracto de la documentación oficial del Framework ZK, traducido y ampliado por Francisco Ferri. Colaborador de Potix (creadores del Framework ZK). Si quieres contactar con él puedes hacerlo en **franferri@gmail.com**, en Twitter **@franciscoferri** o en LinkedIn: **http://www.linkedin.com/in/franciscoferri.**

Autor



Francisco Ferri
Colaborador de Potix (ZK Framework)
Jefe de Proyecto Freelance
Consultor Freelance
Creador de TomyJ Server
Twitter: @franciscoferri
franferri@gmail.com
franferri@javahispano.org



Introducción a ZK y su relacion con RIA (Parte 4)

ZK es un framework de aplicaciones web en AJAX, software de código abierto que permite una completa interfaz de usuario para aplicaciones web



¿Qué son MVC y MVVM?

Son patrones de diseño o modelos de abstracción utilizados para definir y estructurar los componentes necesarios en el de desarollo de software.

Una manera muy simple de describirlos sería:

MVC significa Modelo Vista Controlador, porque en este patrón de diseño se separan los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos. Cuando la lógica de negocio realiza un cambio, es necesario que ella sea la que actualiza la vista.

MVVM significa Modelo Vista VistaModelo, porque en este patrón de diseño se separan los datos de la aplicación, la interfaz de usuario pero en vez de controlar manualmente los cambios en la vista o en los datos, estos se actualizan directamente cuando sucede un cambio en ellos, por ejemplo si la vista actualiza un dato que está presentando se actualiza el modelo automáticamente y viceversa.

Comparación entre MVC y MVVM

La primera imagen describe la interacción que se produce con MVC en ZK, y la siguiente es la correspondiente al enfoque MVVM que hace ZK.

Las principales diferencias entre MVC y MVVM son que en MVVM el “Controller” cambia a “ViewModel” y hay un “binder” que sincroniza la información en vez de hacerlo un controlador “Controller” como sucede en MVC.

MVC

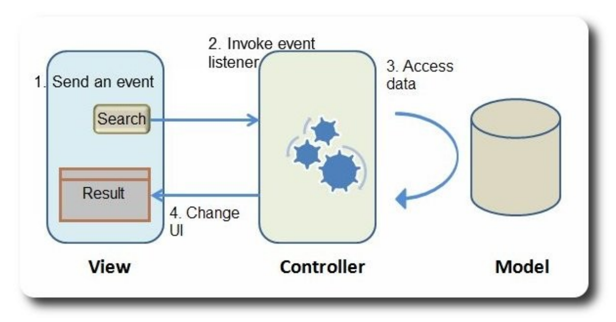


Figura 1.

MVVM

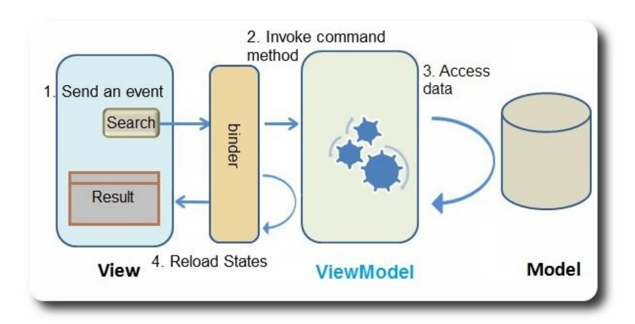


Figura 2.

Los dos enfoques tienen muchas cosas en común, pero también hay claras diferencias entre ellos. Cada uno de los 2 enfoques tiene su razón de ser/existir.

¿Cuál debo usar?

Construir una aplicación mediante MVC puede resultar más intuitivo, porque directamente controlas lo que ves en la vista y su comportamiento, es decir manualmente. MVC se caracteriza porque tienes control total de los componentes, por lo tanto puedes crear componentes hijo dinámicamente (“child”), controlar componentes propios personalizados, o realizar cualquier cosa sobre el componente que este pueda hacer por sí mismo.

En el patrón MVVM, puesto que la capa “ViewModel” esta débilmente acoplada con la vista (no está referenciada a los componentes de la vista), podemos usarla con múltiples vistas sin tener que modificarla. Por lo tanto los diseñadores y programadores pueden trabajar en paralelo. Si la información y comportamiento no cambian, un cambio en la vista no provoca que se tenga que modificar la capa “ViewModel”. Además, como la capa “ViewModel” es un POJO, es fácil realizar tests unitarios sobre ella sin ninguna configuración ni entorno especial. Lo que significa que la capa “ViewModel” tiene una mejor reusabilidad, testabilidad, y los cambios en la vista le afectan menos.

Para sumarizar, comparamos los 2 patrones de diseño en una tabla:

Elemento	MVC	MVVM
Acoplamiento con la vista	Muy poco con plantilla	Muy poco
Acoplamiento con el componente	Un poco	Muy poco
Codificar en la vista	Mediante el ID del componente	A través de una expresión Data binding
Implementación de un controlador	Extendemos ZK's Composer	Es un POJO
Acceso a la información de la UI	Acceso directo	Automático
Acceso a la información desde el backend	Acceso directo	Acceso directo
Actualización de la interfaz de usuario	Manipulamos directamente los componentes	Automático (@NotifyChange)
Nivel de control del componente	Elevado, control total	Normal
Rendimiento	Alto	Normal

Tabla 1.

Acerca de este documento

Este documento es un extracto de la documentación oficial del Framework ZK, traducido y ampliado por Francisco Ferri. Colaborador de Potix (creadores del Framework ZK). Si quieres contactar con él puedes hacerlo en **franferri@gmail.com**, en Twitter **@franciscoferri** o en LinkedIn: **http://www.linkedin.com/in/franciscoferri**

Referencias

[1] Página web oficial de ZK:

http://www.zkoss.org/

[2] MVC en Wikipedia:

http://en.wikipedia.org/wiki/Model-view-controller

[3] MVVM en Wikipedia:

http://en.wikipedia.org/wiki/Model_View_ViewModel

Autor



Francisco Ferri
Colaborador de Potix (ZK Framework)
Jefe de Proyecto Freelance
Consultor Freelance
Creador de TomyJ Server
Twitter: @franciscoferri
franferri@gmail.com
franferri@javahispano.org



Play Web Framework (Parte 3)

En el mundo de desarrollo de software, el uso de los frameworks se ha convertido en una herramienta poderosa, eficiente y eficaz al momento de afrontar proyectos de desarrollo de software, por su fácil aprendizaje, rápido desarrollo y estructura robusta.



Finalizando los estamentos sobre Play! framework continuamos con un tema infaltable en el desarrollo de aplicaciones interactivas, Ajax.

Requisitos

Para el desarrollo de la aplicación se requieren los siguientes elementos:

- ✔ Java Development Kit 1.6.x (OpenJDK u Oracle JDK)
- ✔ Terminal de comandos, Bash(Unix), Command o PowerShell(Windows)
- ✔ Play 1.2.5
- ✔ Acceso a una terminal de comandos, Bash en Unix/Linux o Command en Windows.

Instalación

1. Descargue Play 1.2.5 y descomprímalo en una carpeta (por ejemplo /opt/play12).
2. Agregue la carpeta descomprimida al PATH de su consola de comandos
3. Unix: Agregar la línea export `PATH=$PATH:/opt/play12` a su archivo `~/.bashrc`
4. Windows: Agregar el directorio a través de su utilidad de asignación de variables globales.

Ajax, HTML5, CSS 3 y mas

Retornando en el tiempo a los años noventa vemos la creciente necesidad de aplicaciones y sitios web debido al boom de la burbuja tecnológica. Durante este período se dieron muchos avances que permitieron mejorar y cambiar el enfoque del desarrollo de software para entornos web.

Empezando con Microsoft introduciendo IFrame y los principios de renderizado parcial de páginas hasta HTML 5, tenemos un gran avance en la estandarización y estabilización de un entorno agnóstico de implementación para el desarrollo de software. Actualmente es muy posible tener una aplicación consistente y multiplataforma con solamente seguir los lineamientos definidos por HTML 5.

Ajax es un acrónimo para describir todas las tecnologías usadas en el desarrollo web, que engloban los siguientes elementos:

- ✔ Contenido estático: HTML markup language.
- ✔ Estilos visuales: CSS 3
- ✔ Comportamiento dinámico en los clientes web: Javascript
- ✔ Intercambio de datos entre servidor-cliente: XML y JSON

Siendo estos los bloques fundamentales para el desarrollo web, no siempre presentan las mejores opciones para su desarrollo. Es por

eso que existen varios frameworks web que permiten reducir esta brecha de complejidad entre el estándar y aplicaciones reales en un contexto de negocio.

Contenido dinámico

Rememorando lo dicho en el anterior artículo, tenemos el ciclo de vida de una página en Play!, la cual pasa por los siguientes componentes:

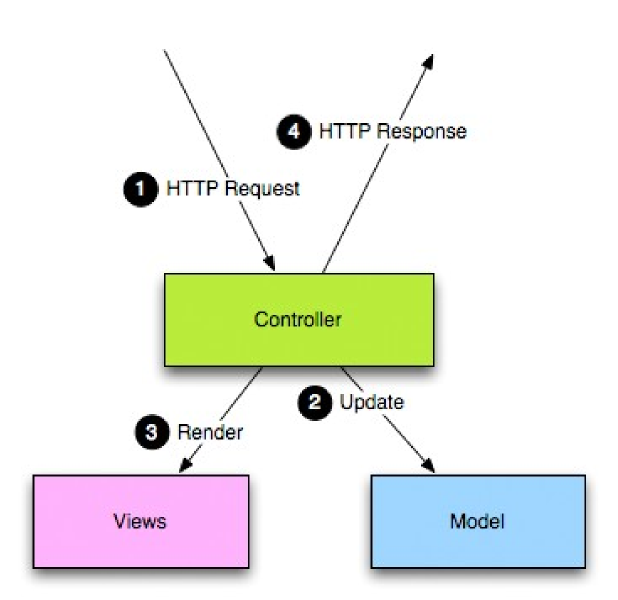


Figura 1.

Este ciclo de vida es dinámico y puede usarse de forma indistinta con cualquier tipo de petición HTTP, he aquí su gran potencial para aplicaciones Ajax. Ya que con estos componentes se puede interactuar de forma transparente con las facilidades proveídas por Play! .

Ajax con Play!

Play! utiliza la librería de Javascript jQuery a través de un tag personalizado jsAction para interactuar de forma simple con los controladores.

jsAction: Retorna una función Javascript que construye una URL basada en una acción del servidor, la cual es ejecutada en un controlador. Adicionalmente permite definir variables de forma libre. Internamente soporta todas operaciones HTTP (GET, PUT, POST, DELETE).

Por ejemplo, teniendo la siguiente ruta mapeada al controlador Usuarios:

```
GET     /usuarios/list              Usuarios.list
```

Podemos ejecutarla usando el tag jsAction con el siguiente código:

```
<script type="text/javascript">
      var listAction = #{ jsAction @list( ':search' , ':size' ) /}
      $( '#result' ).load(
            listAction( { search: 'timoteo', size: '10' } ),
            function(){
                  $( '#content' ).css( 'visibility' ,'visible' )
            }
      )
</script>
```

Y la URL resultante es:

```
GET /usuarios/list?search=timoteo&size=10
```

El fragmento de código Javascript anterior realiza las siguientes operaciones:

1. Crea una función listAction que ejecuta la operación list en el servidor usando los parámetros search y size.
2. Agrega una función sobre el elemento result, llamando a la función listAction colocando los valores de los parámetros previamente definidos.
3. Finalmente agrega una función de retorno que actualiza el elemento content haciéndolo visible cuando retorne el resultado de la operación GET.

Ejemplo

Para nuestro caso de ejemplo usaremos el ejercicio de Tareas que hemos realizado anteriormente debido a su simplicidad.

Primeramente creamos la estructura de un nuevo proyecto con:

```
$> play new TareasAjax
$> cd TareasAjax
```

Creamos una entidad persistente en **`apps/models/Tarea.java`**:

```
package models;

import play.*;
import play.db.jpa.*;

import javax.persistence.*;
import java.util.*;

@Entity
public class Tarea extends Model{

        public String titulo;
        public boolean completada;

        public Tarea(String titulo){
                this.titulo = titulo;
        }
}
```

Configuramos una base de datos en memoria modificando el archivo '**`conf/application.conf`**' con:

```
db = mem
```

Esta configuración define una base de datos en memoria la cual nos permite trabajar de forma local sin necesidad de un sistema de base de datos externo, aunque sus datos son perdidos al reiniciarse la aplicación.

Editamos el controlador principal de la aplicación en **`apps/controllers/Application.java`**:

```
package controllers;

import play.*;
import play.mvc.*;
import java.util.*;
import models.*;

public class Application extends Controller {

    public static void index() {
        List tareas = Tarea.find("ORDER BY id DESC").fetch();
        render( tareas );
    }

    public static void crearTarea( String titulo ){
        Tarea tarea = new Tarea( titulo ).save();
        renderJSON( tarea );
    }

    public static void cambiarEstado(Long id, boolean completada){
        Tarea tarea = Tarea.findById( id );
        tarea.completada = completada;
        tarea.save();
        renderJSON( tarea );
    }
}
```

Este controlador MVC define las siguientes operaciones:

- ✔ **index()**, operación principal que obtiene todas las Tareas persistidas en base a un filtro, y las pone a disposición del motor de plantillas a través de los '**renderArgs**', finalmente recarga la página al llamar al método 'render()' .
- ✔ **crearTarea()**, permite crear tareas retornando un resultado formateado como JSON para su posterior renderizado.
- ✔ **cambiarEstado()**, permite cambiar el estado de tareas existentes actualizando su bandera de completada, también retorna un resultado como JSON para su interacción con Ajax.

Y un archivo más con la ruta '**app/views/Application/index.html**' con el siguiente contenido:

```
#{extends 'main.html' /}
#{set title:'Tareas' /}

#{ifnot tareas}
    <p>
        No hay tareas.
    </p>
#{/ifnot}

<ul>
    #{list items: tareas, as: 'tarea'}
        <li>
            <input type="checkbox" id="${tarea.id}"
                ${tarea.completada ? 'checked' : ''}/>
            ${tarea.titulo}
        </li>
    #{/list}
</ul>
```

```
<p>
    <a id="crearTarea" href="#"> Crear una nueva tarea </a>
</p>

<script type="text/javascript">
    //crear una tarea
    $( '#crearTarea' ).click( function(){
        $.post( '@{crearTarea()}' ,
            {titulo: prompt( 'Titulo de la tarea?' )},
            function( tarea ){
                $('ul').prepend(
             '<li><input type="checkbox" id="' +
                                tarea.id + '"/>' +
                                tarea.titulo +
                    '</li>'
                )
            },
            'json'
        )
    })

    //cambiar el estado de la tarea
    $('input').live( 'click', function(){
        $.post( '@{cambiarEstado()}',
            {id: $(this).attr('id'), completada: $(this).val()}
        )
    })

</script>
```

El fragmento de código anterior utiliza el motor de plantillas descrito en el anterior artículo, pero adicionalmente hace uso de código Javascript, revisémoslo:

Crear nueva tarea, en este fragmento de código se llama a una función crearTarea usando un título ingresado por el usuario. Adicionalmente se agrega una función callback que agrega la tarea retornada en formato JSON al contenido actual.

Cambiar estado de la tarea, esta operación llama a la función cambiarEstado usando el id y valor del checkbox actual, esto no recarga la página ni sus datos, pero envía una señal al servidor para que actualice tal entrada.

Ahora inicie la aplicación con:

```
$> play run
```

y con el navegador web ingrese a **`http://localhost:9000/`**, tendrá el siguiente resultado:

- ☐ Tarea 4
- ☐ Tarea 5
- ☐ Tarea 3
- ☐ Tarea 2
- ☐ Tarea 1

Crear una nueva tarea

Figura 2.

Proceda con las siguientes acciones:

- ✔ Agregue nuevas tareas usando en enlace, verá que las tareas son agregadas sin necesidad de recargar la página.



Figura 3.

- ✔ Cambie el estado de las tareas a través del checkbox, se dará cuenta que al refrescar la página sus valores no se pierden, ya que esta actualizando la base de datos a través de Ajax.

- ☐ Tarea 7
- ☐ Tarea 7
- ☑ Tarea 6
- ☑ Tarea 4
- ☑ Tarea 5
- ☑ Tarea 3
- ☑ Tarea 2
- ☑ Tarea 1

Crear una nueva tarea

Figura 4.

Consideraciones importantes

- ✔ Nuevas restricciones en HTML 5 prohíben interacciones Javascript entre sitios ajenos (cross-site scripting).
- ✔ Todas las peticiones HTTP son respondidas por los controladores si es que ninguna regla de seguridad es agregada.
- ✔ El renderizado parcial de páginas no es controlado por el framework, esto quiere decir que el desarrollador debe definir que partes actualizar durante las interacciones Ajax.
- ✔ Las operaciones POST son preferibles sobre operaciones GET debido a que los datos de formulario pueden enviarse encriptados.

Quiere usar un IDE?

- ✔ **play ant**: Genera un archivo de construcción Ant para la aplicación.
- ✔ **play eclipsify**: Genera los archivos de configuración para Eclipse.
- ✔ **play netbeansify**: Genera los archivos de configuración para Netbeans.

- ✔ **play idealize**: Genera los archivos de configuración para IntelliJ.

Qué fué lo que hicimos?

Recopilemos lo que hicimos para nuestra aplicación Tareas:

1. Creamos la aplicación base.
2. Agregamos el elemento de dominio Tarea usando el componente base Model.
3. Habilitamos una base de datos en memoria a través de la configuración db=mem
4. Modificamos el componente controlador base denominado Application
5. Modificamos la plantilla principal de la aplicación en '`app/views/Application/index.html`'
6. Agregamos contenido estático y comportamiento dinámico a la página principal usando el tag jsAction y comunicación asíncrona usando JSON.

Conclusiones

Play! ofrece una pila completa de componentes y librerías que conforman una plataforma de desarrollo web de fácil uso y baja curva de aprendizaje, en comparación a muchos otros frameworks web en el mercado.

Desde la persistencia de datos, pasando por el procesamiento de rutas y URLs hasta la interacción asíncrona con los browsers cliente a través de sus utilidades Ajax. Play! resalta como un framework equivalente a Rails y Django resolviendo muchos de los problemas de escalabilidad que tienen estos al estar subrayado por las bondades ofrecidas en un entorno tan robusto como Java.

En nuevas versiones la comunidad de Play! ha dado un paso de gigante al agregar soporte de Scala para el desarrollo, restructurando completamente su framework hacia este lenguaje 100% compatible con la máquina virtual de Java y que ofrece bondades equivalentes a las de Ruby con respecto a simplificación de código y programación objeto-funcional.

En conclusion final, Play! no puede obviarse en el desarrollo de software, es extremadamente útil para realizar proyectos prototipo y reales, su uso es simplificado e integrado con las herramientas de desarrollo de más importancia en el momento.

Enlace a todos los artículos y su código fuente: **`https://github.com/timoteoponce/articles`**

Referencias

[1] http://www.playframework.org/documentation/1.2.5/home

[2] http://en.wikipedia.org/wiki/HTML5

[3] http://en.wikipedia.org/wiki/Ajax_%28programming%29

[4] http://www.playframework.com/documentation/1.2/ajax

Autor



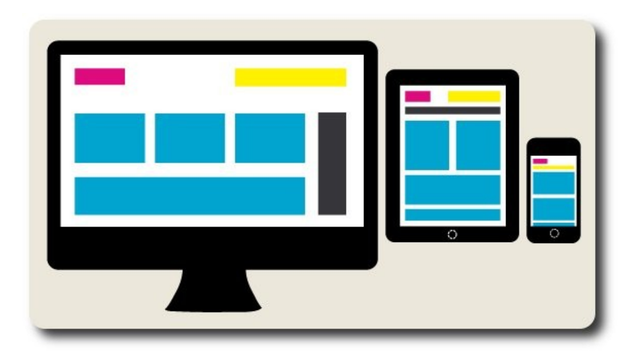
Timoteo Ponce

Ingeniero de Software – Consultor Técnico

timo@gmail.com

RESS: Responsive Design + Server Side Components

Un panorama del desarrollo de aplicaciones web para dispositivos móviles, donde incluso en las interfaces de usuario modo texto podemos elaborar aplicaciones elegantes y llamativas.



Introducción

La situación actual demuestra que el acceso a la web se realiza mediante variedad de dispositivos, como ser: televisión, tablets y sobre todo dispositivos móviles, los cuales continúan con el incremento de su popularización, de tal manera que las organizaciones están dándose cuenta de la necesidad de invertir recursos en el desarrollo de aplicaciones web que estén orientados a la adaptación de la diversidad de dispositivos existentes.

Las acciones, para la concepción de aplicaciones web adaptables, generalmente encuentran un primer inconveniente al momento de realizar la elección de un método adecuado para lograr este fin. Entre las alternativas se encuentran el diseño web sensible, la generación de versiones distintas de acuerdo al dispositivo, y las soluciones basadas en la detección de características del lado del servidor.

El presente artículo da a conocer una perspectiva sobre las soluciones basadas en el lado del servidor, de tal modo que se constituya en una alternativa al momento de desarrollar soluciones orientadas a múltiples dispositivos.

Detección de características

Con la gran variedad de dispositivos móviles existentes, se hace necesario contar con mecanismos que permitan identificar sus características, de modo que, los sitios web se puedan desplegar de manera más óptima y de acuerdo a las características de los navegadores clientes, en ese sentido existen los siguientes métodos que permiten llevar a cabo esta actividad (Wroblewski, 2012):

- ✔ **Responsive Web Desing (RWD)**: Se basa en la aplicación de diseño e imágenes fluidas, así como de "media queries" para determinar la detección de características del navegador en el lado del cliente para de esa manera proporcionar una versión adecuada del sitio.
- ✔ **Device experiences**: Está determinado por la manera en que un dispositivo es utilizado y dependiendo de las capacidades técnicas que tiene. Generalmente consiste en el diseño del front-end orientado a cada clase de dispositivo objetivo y enviar al cliente solo lo que sea necesario.
- ✔ **Responsive Design and Server Side Components (RESS)**: Hace referencia al RWD tomando en cuenta componentes del lado del servidor.

Detección de características del lado del servidor

Las primeras versiones de los dispositivos móviles no tenían capacidad para soportar CSS ni javascript por lo cual no se podía redefinir el contenido a ser desplegado; a esto se adhería la limitación del peso de las páginas, de manera que para lograr que éstas se carguen, el contenido no debía ser muy extenso ni incluir elementos gráficos o multimedia. Por lo tanto, para librar estas restricciones solo quedaba "armar" el contenido de acuerdo a las capacidades de los dispositivos cliente.

Aunque con la llegada de los Smartphones, muchas de esas limitaciones han sido superadas, aún se pueden identificar algunas situaciones donde el RWD no se muestra como una solución absoluta:

- ✔ **Adaptación del contenid**o: Desplegar contenido diferenciado, por ejemplo si se está navegando en Android o iOS, solo se debería mostrar el store o los recursos pertenecientes al sistema operativo que se está haciendo uso.
- ✔ **Rendimiento**: Resulta complicado proporcionar un sitio óptimo usando solamente adaptaciones del lado del cliente porque el servidor envía el mismo contenido sea cual sea el sistema operativo que está realizando la petición.
- ✔ **Dispositivos muy antiguos**: Algunos dispositivos solo soportan un subconjunto del HTML lo que imposibilita que un sitio web sea desplegado con todas sus características.
- ✔ **Nuevos dispositivos**: Actualmente la televisión está siendo considerada para interactuar con elementos web, sin embargo su personalización aún no se encuentra optimizada, por ejemplo no soportan el reconocimiento del media "tv".

Mejorar estos aspectos requiere que se pueda realizar la detección de características desde el lado del servidor y permitir liberar contenido orientado a las características específicas de los dispositivos.

RESS se constituye en el método por el cual se puede realizar el descubrimiento de estas características, obteniéndose también a partir de su aplicación, las siguientes ventajas:

- ✔ **Velocidad y eficienci**a: El servidor solo prepararía el contenido específico necesario, acortando los tiempos de carga de las páginas.
- ✔ **Precisión de información**: Informa de manera exacta los tipos de media que soporta, los formatos de imágenes, el códec de video disponible, etc.
- ✔ **Solución única**: Se logra construir un solo sitio orientado a varios dispositivos.

¿AJAX?
¿orientación?
¿HTML5?
¿wifi?
¿resolución de pantalla?
¿tipo de navegación?
¿gif animados?
¿sistema operativo?
¿CSS3?
¿flash?
¿tipo de streaming?
¿j2me?

Figura 1.

Métodos de detección del lado del servidor

Entre los más populares se puede hacer referencia:

- ✔ **Device detection (Detección de dispositivo)**: Consiste en él envío de cabeceras HTTP por parte el navegador hacia el servidor cuando alguna página o recurso es requerido. Su objetivo es revelar las características del navegador que se

están haciendo uso. La información que se obtiene puede resultar bastante ampulosa, por lo cual existen herramientas que facilitan el tratamiento de los mismos, las cuales son conocidas como: Device Detection Repositories (DDR).

Un DDR consiste en un repositorio de información sobre las características (navegador, sistema operativo, etc.) que soportan diversidad de dispositivos móviles. Mediante esta información se puede optimizar la experiencia de los usuarios, enviando solamente los elementos necesarios al dispositivo.

- **Feature detection (Detección de características)**: Se encuentra basado en la detección del lado del cliente, usa javascript para determinar que características son soportadas por el navegador del usuario.

  Para llevara a cabo la detección de características en el lado del cliente se puede hacer uso de herramientas especializadas como Modernizr **http://modernizr.com**; sin embargo para optimizar el rendimiento desde el lado del servidor, es necesario que éste conozca la información antes de que el recurso sea enviado al navegador, pensando en ello James Pearce ha creado **modernizr-server**, cuya librería y guía de aplicación se puede encontrar en la siguiente dirección **https://github.com/jamesgpearce/modernizr-server**

Herramientas DDR

Estas herramientas por lo general constan de dos componentes fundamentales:

- Una base de datos de información de los dispositivos.
- Una API que permite asociar un requerimiento (generalmente HTTP) a la lista de propiedades.

Entre los DDR más populares se encuentran:

- WURFL **http://www.scientiamobile.com/**
- DeviceAtlas **https://deviceatlas.com/**
- 51Degrees **http://51degrees.mobi/**
- OpenDDR **http://www.openddr.org/**
- DetectRight **http://www.detectright.com/**
- Apache Mobile Filter **http://www.apachemobilefilter.org/**

Estas soluciones consisten en un gran repositorio de información que ayuda a reconocer las diferentes capacidades de los dispositivos. Por capacidades se hace referencia a la habilidad para soportar ciertas características y estándares como ser: formatos de imágenes, etiquetas html, tamaño de pantallas, resolución y paleta de colores soportada por la pantalla, etc.

Estos proyectos tienen sus versiones open-source y están disponibles para su interacción con lenguajes de programación de uso común.

Conclusiones

Aunque el diseño sensible y la detección del lado del servidor suelen ser conceptos y posiciones enfrentadas, ninguna constituye una solución absoluta de manera individual. La aplicación combinada de ambas soluciones puede resultar en una mejor experiencia y encontrarse orientada a muchos dispositivos.

La detección del lado del servidor debe servir para mejorar la experiencia de los usuarios y no restringir su acceso, ya que haciendo uso de esta alternativa por ejemplo se puede lograr que un recurso solo sea accesible mediante cierto tipo de navegador.

La detección de características del lado del cliente es la solución más popular, que sin embargo podría tener un mejor rendimiento si se apoya en el tratamiento del lado del servidor.

Recursos consultados

[1] Kadlec, T. (2013). Implementing Responsive Desing. New Riders.

[2] Wroblewski. (29 de 02 de 2012). Lukew Idertion + Design. Obtenido de http://www.lukew.com/ff/entry.asp?1509

[3] Wroblewski, L. (29 de febrero de 2012). Which One: Responsive Design, Device Experiences, or RESS? Obtenido de http://www.lukew.com/ff/entry.asp?1509

Autor



Marco Antonio Avendaño Ajata

marcoviaweb@gmail.com

Biblioteca Estándar



Características de la biblioteca estándar

Esta sección de la guía PyMOTW incluye introducciones a varios módulos de la biblioteca estándar basadas en características, organizadas por el que podría ser tu objetivo. Cada artículo puede incluir referencias cruzadas a varios módulos de diferentes partes de la biblioteca, y mostrar cómo se relacionan entre sí.

Persistencia e intercambio de datos

Python ofrece varios módulos para almacenar datos. Hay básicamente dos aspectos de persistencia: convertir el objeto en la memoria de ida y vuelta en un formato para guardarlo, y trabajar con el almacenamiento de los datos convertidos.

Serializando objetos

Python incluye dos módulos capaces de convertir objetos en un formato transmisible o almacenable (serializar) pickle y json. Es más común usar pickle, ya que existe una implementación rápida en C y está integrado con algunos de los otros módulos de la biblioteca estándar que almacenan los datos serializados, como el módulo shelve. Aplicaciones Web querrán sin embargo examinar json, ya que se integra mejor con algunas de la aplicaciones Web de almacenamiento existentes.

Guardando objetos serializados

Una vez que el objeto en la memoria es convertido en un formato almacenable, el siguiente paso es decidir cómo almacenar los datos. Un simple archivo plano con objetos serializados escritos uno tras otro funciona con datos que no necesitan ser indexados de alguna manera. Pero Python incluye una colección de módulos para almacenar pares llave-valor en una base de datos simple usando una de las variantes del formato DBM.

La interfaz más simple para aprovechar el formato DBM es proporcionada por shelve. Simplemente abre el archivo shelve y accede a él a través de interfaz como diccionario. Objetos guardados en el shelve son automáticamente convertidos y guardados sin ningún trabajo adicional de tu parte.

Un inconveniente de shelve es que con la interfaz por defecto no se puede garantizar qué formato de DBM se utilizará. Eso no importará si u aplicación no necesita compartir los archivos de base de datos entre equipos con diferentes bibliotecas, pero si eso es necesario, puedes usar una de las clases en el módulo para garantizar que un formato específico es seleccionado.

Si vas a estar pasando una gran cantidad de datos a través de JSON de todas maneras, usando json y anydbm pueden proporcionar otro mecanismo de persistencia. Dado que las llaves de la base de datos DBM y los valores deben ser cadenas, sin embargo, los objetos no se recrean automáticamente cuando se accede a los valores en la base de datos.

Bases de datos relacionales

El excelente sqlite3, base de datos relacional, está disponible con la mayoría de las distribuciones de Python. Almacena la base de datos en la memoria o en un archivo local y todo el acceso es desde dentro del mismo proceso, por lo que no hay ningún retraso por la red La naturaleza compacta de sqlite3 hace que sea especialmente adecuado para ser integrado en una aplicaciones de escritorio o versiones de desarrollo de aplicaciones Web.

Todo acceso a la base de datos es a través de la interfaz de Python DBI 2.0, por defecto, ya que no está incluido un mapeador de objetos relacional (ORM). El mapeador de propósito general más popular es SQLAlchemy, pero otros como el de mapeador nativo de Django también soportan SQLite. SQLAlchemy es fácil de instalar y configurar, pero si los objetos no son muy complicados y tú está preocupado acerca de sobrecarga, es posible que desees utilizar la interfaz DBI directamente.

Intercambio de datos a través de formatos estándar

Aunque no es generalmente considerado un verdadero formato de persistencia, archivos csv (valores separados por coma) pueden ser una manera efectiva de migrar datos entre aplicaciones. La mayoría de los programas de hojas de cálculo y bases de datos soportan la exportación e importación de datos usando CSV, por lo que hacer dump de datos a un archivo CSV e con frecuencia la manera más simple de mover datos de tu aplicación a una herramienta de análisis.

Estructuras de datos en la memoria

Python incluye varias estructuras de datos estándar de programación como tipos incorporados (lista, tupla, diccionario y conjunto. La mayoría de las aplicaciones no van a necesitar otras estructuras, pero cuando lo hacen, la biblioteca estándar tiene lo necesario.

array

Para grandes cantidades de datos, puede ser más eficiente usar un array en lugar de una list. Puesto que la matriz está limitada a un sólo tipo de datos, puede usar una representación más compacta de memoria que una lista de propósito general. Como beneficio adicional, matrices pueden ser manipuladas usando muchos de los mismo métodos que una lista, por lo que puede ser posible reemplazar listas con matrices en tu aplicación sin muchos cambios.

Ordenando

Si necesitas mantener una lista ordenada a medida que añades y quitas valores, echa un vistazo a heapq. Usando las funciones de heapq para añadir o quitar elementos de una lista, puedes mantener el orden de la lista con bajo consto operativo.

Otra opción para crear listas ordenados o matrices es bisect. Bisect utiliza una búsqueda binaria para encontrar el punto de inserción para nuevos elementos, y es una alternativa a ordenar repetidamente una lista que cambia con frecuencia.

Queue

Aunque la lista incorporada puede simular una cola utilizando los métodos insert() y pop(), no es seguro para subprocesos. Para una comunicación ordenada de verdad entre hilos de ejecución deberías usar Queue. multiprocessing incluye una versión de Queue que funciona entre procesos, haciendo más fácil portar entre los módulos.

collections

collections incluye implementaciones de varias estructuras de datos que extienden las de otros módulos. Por ejemplo, Deque es una cola de doble terminación y te permita añadir o quitar elementos de ambos extremos. El defaultdict es un diccionario que responde con un valor pre determinado si la llave no se encuentra en el diccionario. Y namedtuple extiende la tupla normal para dar a cada miembro un atributo nombre además de un índice numérico.

Decodificando datos

Si estás trabajando con datos de otra aplicación, tal vez procedente de un archivo binario o una secuencia de datos, encontrarás struct útil para decodificar los datos a tipos nativos de Python para facilitar la manipulación.

Variaciones personalizadas

Y por último, si los tipos disponibles no te dan lo que necesitas, es posibles que desees hacer una sub clase de uno de los tipos nativos y personalizarlo. También puedes empezar utilizado las clases bases abstractas en collections.

Acceso a archivos

La biblioteca estándar de Python incluye una amplia gama herramientas para trabajar con archivos, con nombres de archivos y el contenido de archivos.

Nombres de archivos

El primer paso para trabajar con archivos es obtener el nombre del archivo para que puedas operar en él. Python representa a los nombres de archivos como cadenas simples, pero proporciona herramientas para su creación a partir de componentes estándar, independientes de la plataforma en os.path. Lista el contenido de un directorio con listdir() de os, o utiliza glob para crear una lista de nombres de archivos en base a un patrón. Un filtrado más fino de nombres de archivos es posible con fnmatch`.

Meta-datos

Una vez que conoces el nombre de un archivo, es posible que quieras revisar otras características como los permisos o el tamaño usando os.stat() y las constantes en stat.

Leyendo archivos

Si estás escribiendo una aplicación filtro que procesa la entrada de texto línea por línea, fileinput proporciona un marco sencillo para empezar. La interfaz de programación te pide iterar sobre el generador input(), procesando cada línea a medida que se produce. El generador se encarga del análisis de los argumentos de la línea de comando para nombres de archivos, o termina leyendo directamente de sys.stdin. El resultado es una herramienta flexible que tus usuarios pueden ejecutar directamente en un archivo o como parte de una pipeline.

Si tu aplicación necesita acceso aleatorio a archivos, linecache facilita leer líneas por su número de línea. El contenido del archivos es mantenido en un cache, así que ten cuidado del consumo de memoria.

Archivos temporales

Para casos en que necesites crear archivos para almacenar datos temporalmente, o antes de moverlos a una ubicación permanente, tempfile será muy útil. Ofrece clases para crear archivos temporales y directorios de forma segura y confiable. Se garantiza que los nombres no colisionan e incluyen componentes aleatorios, que los hace difíciles de adivinar.

Archivos y directorios

Con frecuencia es necesario trabajar con un archivo, sin preocuparse de su contenido. El módulo shutil incluye operaciones de alto nivel en archivos como copiar archivos y directorios, configurar los permisos, etc.

Herramientas de procesamiento de texto

La clase string es la herramienta de procesamiento de texto más obvia disponible para programadores de Python, pero hay un montón de otras herramientas en la biblioteca estándar para hacer más simple la manipulación de texto.

Módulo string

Código de estilo viejo utiliza funciones del módulo string, en lugar de métodos de objetos string. Hay un método equivalente para cada función del módulo, y el uso de las funciones ha quedado en desuso para código nuevo.

Código nuevo puede utiliza un string.Template como una manera simple de parametrizar cadenas más allá de las características de las cadenas y las clases unicode. Aunque no es tan rica en características como plantillas definidas por muchos de frameworks de Web o módulos de extensión disponibles en

PyPI, string.Template es un buen término medio para plantillas modificables por el usuario donde valores dinámicos necesitan ser insertados en texto de otro modo estático.

Texto de entrada

Leer de un archivo es bastante fácil, pero si estás escribiendo un filtro línea por línea, el módulo fileinput es aún más fácil. La interfaz de fileinput requiere que iteres sobre el generador input(), procesando cada línea a medida que se produce. El generados se encarga de analizar los argumentos de la línea de comando como nombres de archivos, o lee directamente de sys.stdin. El resultado es es una herramienta flexible que tus usuarios pueden ejecutar directamente en un archivo o como parte de una pipeline.

Texto de salida

El módulo textwrap incluye herramientas para dar formato a texto desde árrafos limitando la anchura de la salida, adicionando la sangría, e insertando saltos de línea para ajustar las líneas de forma coherente.

Comparando valores

La biblioteca estándar incluye dos módulos relacionados con la comparación de valores de texto más allá de la igualdad incorporada y la comparación de orden con el apoyo de objetos string. re proporciona una biblioteca completa de expresiones regulares, implementada en C por razones de rendimiento. Expresiones regulares son muy adecuadas para encontrar sub cadenas dentro de un conjunto más grande de datos, comparando cadenas con un patrón (en lugar de otra cadena fija) y el análisis sintáctico suave.

difflib, por otra parte, te muestra las diferencias reales entre secuencias de texto en términos de partes añadidas, borradas, o cambiadas. La salida de las funciones de comparación en difflib puede ser usada para proporcionar información más detallada al usuario acerca de dónde ocurren cambios en dos entradas, cómo un documento ha cambiado con el tiempo, etc.

Autor



Ernesto Rico Schmidt

Usuario de Linux y Software Libre desde 1994
e.rico.schmidt@gmail.com



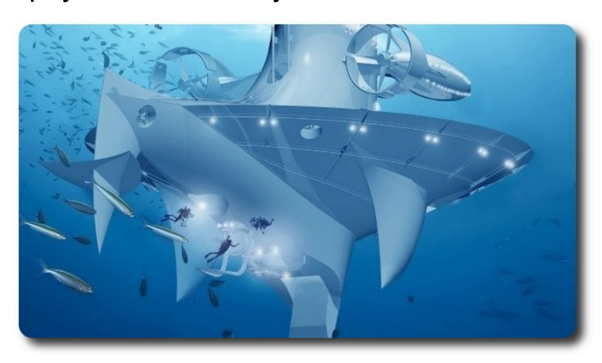

Willay
News
GNU/LINUX
TECNOLOGÍA
CONOCIMIENTO

Novedades Tecnológicas

SeaOrbiter

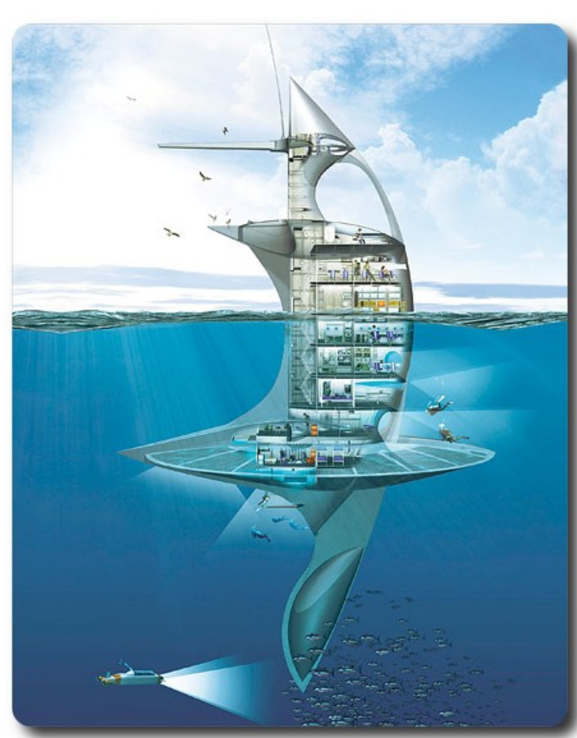


El SeaOrbiter, también conocido como Sea Orbiter (dos palabras), es un buque de investigación oceánica que hará su viaje inaugural en 2013. Al igual que una nave espacial, se planea que el SeaOrbiter provea a científicos y a otras personas una estación móvil de investigación residencial bajo la superficie de los océanos. La estación dispondrá de laboratorios, talleres, habitaciones y una terraza a presión para apoyar buceadores y submarinos.



El SeaOrbiter es un proyecto de la organización del «laboratorio oceanográfico flotante». Está dirigido por el arquitecto francés Jacques Rougerie, el oceanógrafo Jacques Piccard y el astronauta Jean-Loup Chrétien. La construcción del laboratorio oceanográfico comenzó en 2011. Se espera que el costo sea alrededor de $52,7 millones.

El laboratorio es una embarcación oceánica semisumergible y pesa 1000 toneladas. Tiene una altura total de 51 metros con 31 metros bajo el nivel del mar.

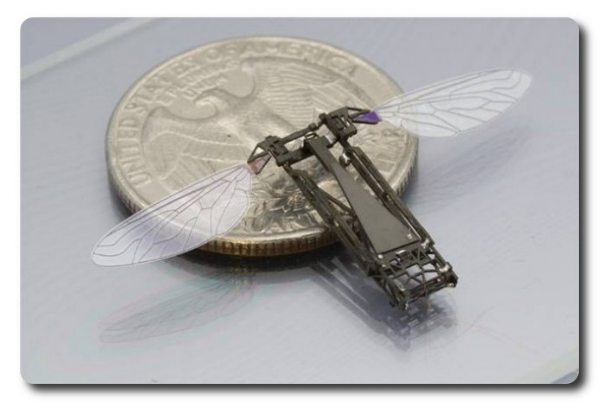


Ha sido diseñado para flotar en posición vertical e ir a la deriva con las corrientes del océano, pero tiene dos hélices pequeñas que le permiten modificar su trayectoria y maniobrar en aguas confinadas. Puede enviarse robots submarinos desde el laboratorio para explorar el fondo del mar. El casco hecho de una aleación de aluminio y magnesio es cinco veces más grueso que la de un buque convencional.

Su alineación vertical en el mar dejará una pequeña parte visible por encima de la superficie (con un alojamiento mucho más grande) y los laboratorios por debajo de la superficie del mar. Algunos niveles tendrán una cabina de presión igual a la presión externa del agua permitiendo a los buzos vivir durante largos períodos en profundidad y hacer frecuentes excursiones.

Robot mosca

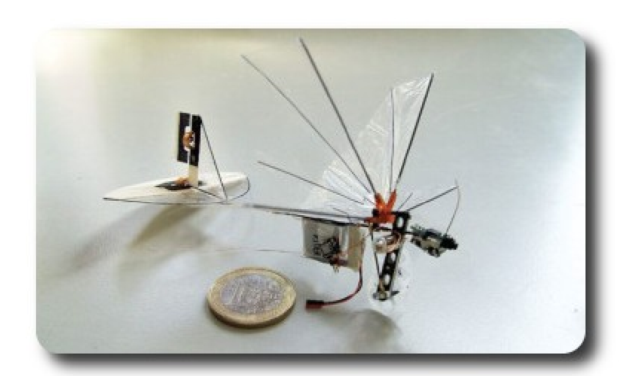
Científicos de la Universidad de Harvard han creado un pequeño robot del tamaño de una mosca casera capaz de volar como ella. De apenas 80 miligramos de peso y 3 cm de envergadura, el pequeño ingenio está fabricado en materiales ultraligeros que le permiten mover sus alas 120 veces por segundo, tan rápido que algunas cámaras no pueden captarlo. Se trata del primer insecto robótico que imita a la perfección a sus análogos naturales y su creación ha llevado más de diez años de investigación y desarrollo.



RoboBee, que aparece descrito esta semana en la revista Science, es el fruto del empeño de los científicos por imitar la velocidad extraordinaria y la capacidad de maniobra de los insectos voladores. Las alas del robot está compuestas de un fino poliester reforzado con fibra de carbono y sus «músculos» son en realidad cristales piezoeléctricos que se contraen o se estiran según el voltaje que se les aplica (Una batería, en una década)

Los componentes son tan pequeños, algunos tan solo miden micrómetros, que los investigadores han tenido que innovar en su fabricación, ya que no era posible utilizar procesos convencionales. Lo que todavía no está bien solucionado es la fuente de energía, ya que, al ser tan ligero (80 miligramos), Robobee no puede llevarla encima, así que tiene que estar controlado desde el suelo. Un ordenador monitoriza sus movimientos y ajusta su altitud.

Con todo, es el primer robot capaz de volar como una mosca, incluidos su revoloteos. Los científicos creen que una batería suficientemente pequeña para que el robot la pueda cargar encima llegará dentro de cinco a diez años.



RoboBee podría ser destinado a operaciones de búsqueda y rescate (por ejemplo, en edificios a punto de derrumbarse) o para ayudar a la polinización de los campos, especialmente en un mundo en el que las auténticas abejas son cada vez menos numerosas.

Autor



Jenny Saavedra López
Diseño y Edición Revista Atix
jenny.saavedra@atixlibre.org

Libres para pensar, libres para decidir, libres para crear



Te ofrecemos este espacio para mostrar tu Creatividad

ATIX

La Fuerza del Software Libre

Envíanos tus diseños y creaciones para publicarlos


Libre
atix
fundación

Contacto

Para solicitar cualquier información, puedes contactar a:

- Esteban Saavedra López (esteban.saavedra@atixlibre.org)
- Jenny Saavedra (jenny.saavedra@atixlibre.org)

Visita nuestro sitio web y descarga todos los números

- http://revista.atixlibre.org

Envío de Artículos

Te invitamos a participar de la Revista Atix enviándonos artículos referidos a las siguientes áreas :

- Instalación y personalización de aplicaciones
- Scripting
- Diseño gráfico
- Programación y desarrollo de aplicaciones
- Administración de servidores
- Seguridad
- y cualquier tema enmarcado dentro del uso de Software Libre
- Conocimiento Libre
- Tecnología Libre
- Cultura Libre
- Trucos y recetas.
- Noticias.


Te falta algun número de
atix
obtenlos de nuestra web
ATIX
Revista Digital
oteger lo nuestro
2011
ATIX
Revista Digital
La Libertad es
ATIX
Revista Digital
El mundo ne
Revista Digital
ATIX
Revista Digital
ATIX
Trabajo y
Tradición
atix
Libre
atix
Libre



Por un Mundo Ético, Libre y Justo

http://revista.atixlibre.org